Classification_Models

November 18, 2025

[1]:

```
# Cell 1: Import libraries and load dataset
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪accuracy_score, make_scorer

# Load dataset
df = pd.read_csv("Android data for classification.csv")

# Display first rows
df.head()
```

[1]:

```
   WMC  DIT  NOC  CBO  RFC  LCOM  Ca  Ce  NPM   LCOM3  LOC     DAM     CAM  \
0    3    1    0    0    4     3   0   0    1  1.5000   24  1.0000  0.5556
1   12    1    0    0   13    66   0   0    9  1.0909  116  0.1136  0.1944
2   17    1    0    2   18   136   0   2   14  1.0625  124  0.9545  0.1674
3    6    1    0    0    7    15   0   0    6  1.2000   44  0.2500  0.4333
4   11    1    0    7   12    55   3   4    8  1.1000   83  0.8235  0.2364

  DEFECT
0    yes
1    yes
2    yes
3    yes
4    yes
```

[2]:

```
# Cell 2: Prepare data and scoring metrics

# Assuming last column is the target. Change if needed.
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Define scorers
scoring = {
    'accuracy': make_scorer(accuracy_score),
    'precision': make_scorer(precision_score, average='macro'),
    'recall': make_scorer(recall_score, average='macro'),
    'f1': make_scorer(f1_score, average='macro')
}

# KFold objects
kfold_5 = KFold(n_splits=5, shuffle=True, random_state=42)
kfold_10 = KFold(n_splits=10, shuffle=True, random_state=42)

X.shape, y.shape
```

[2]:
```
((78, 13), (78,))
```

[4]:
```
# Cell 3 (Corrected): 5-fold Cross Validation

from sklearn.model_selection import cross_validate

models = {
    "Naive Bayes": GaussianNB(),
    "KNN (k=5)": KNeighborsClassifier(n_neighbors=5),
    "Decision Tree": DecisionTreeClassifier(random_state=42)
}

results_5 = {}

for name, model in models.items():
    print(f"----- {name} (5-fold) -----")

    scores = cross_validate(
        model, X, y, cv=kfold_5,
        scoring=scoring,
        return_train_score=False
    )

    results_5[name] = {
        'Accuracy': scores['test_accuracy'].mean(),
        'Precision': scores['test_precision'].mean(),
        'Recall': scores['test_recall'].mean(),
        'F1 Score': scores['test_f1'].mean()
    }

    print("Accuracy:", scores['test_accuracy'].mean())
    print("Precision:", scores['test_precision'].mean())
    print("Recall:", scores['test_recall'].mean())
    print("F1 Score:", scores['test_f1'].mean())
    print()
```

```
----- Naive Bayes (5-fold) -----
Accuracy: 0.6808333333333334
Precision: 0.7184920634920634
Recall: 0.7072619047619048
F1 Score: 0.6770378151260503

----- KNN (k=5) (5-fold) -----
Accuracy: 0.6791666666666666
Precision: 0.6967055167055166
Recall: 0.6620238095238096
F1 Score: 0.6574699878724646

----- Decision Tree (5-fold) -----
Accuracy: 0.63
Precision: 0.6113858363858363
Recall: 0.6173015873015874
F1 Score: 0.6067260448529799
```

[5]:

```python
# Cell 4: 10-Fold Cross Validation

results_10 = {}

for name, model in models.items():
    print(f"----- {name} (10-fold) -----")

    scores = cross_validate(
        model, X, y, cv=kfold_10,
        scoring=scoring,
        return_train_score=False
    )

    results_10[name] = {
        'Accuracy': scores['test_accuracy'].mean(),
        'Precision': scores['test_precision'].mean(),
        'Recall': scores['test_recall'].mean(),
        'F1 Score': scores['test_f1'].mean()
    }

    print("Accuracy:", scores['test_accuracy'].mean())
    print("Precision:", scores['test_precision'].mean())
    print("Recall:", scores['test_recall'].mean())
    print("F1 Score:", scores['test_f1'].mean())
    print()
```

```
----- Naive Bayes (10-fold) -----
Accuracy: 0.6964285714285714
Precision: 0.7202380952380951
```

```
Recall: 0.7252380952380952
F1 Score: 0.6661002886002886

----- KNN (k=5) (10-fold) -----
Accuracy: 0.6696428571428571
Precision: 0.6715476190476191
Recall: 0.6370238095238095
F1 Score: 0.621510989010989

----- Decision Tree (10-fold) -----
Accuracy: 0.5803571428571428
Precision: 0.5353571428571429
Recall: 0.5523809523809524
F1 Score: 0.5225885225885225
```

[6]:

```python
# Cell 5: Create tables for 5-fold and 10-fold results

import pandas as pd

table_5 = pd.DataFrame(results_5).T
table_10 = pd.DataFrame(results_10).T

print("5-Fold Cross Validation Results:")
display(table_5)

print("10-Fold Cross Validation Results:")
display(table_10)
```

```
5-Fold Cross Validation Results:

               Accuracy  Precision    Recall  F1 Score
Naive Bayes    0.680833   0.718492  0.707262  0.677038
KNN (k=5)      0.679167   0.696706  0.662024  0.657470
Decision Tree  0.630000   0.611386  0.617302  0.606726

10-Fold Cross Validation Results:

               Accuracy  Precision    Recall  F1 Score
Naive Bayes    0.696429   0.720238  0.725238  0.666100
KNN (k=5)      0.669643   0.671548  0.637024  0.621511
Decision Tree  0.580357   0.535357  0.552381  0.522589
```

1 Summary of Classification Model Performance

1.1 1. Overview

Three classification algorithms were evaluated on the given dataset:

- **Naive Bayes**

- **K-Nearest Neighbors (KNN, k=5)**
- **Decision Tree**

Each model was tested using both **5-fold** and **10-fold** cross validation. Performance was compared using **Accuracy, Precision, Recall, and F1 Score**.

1.2 2. Key Findings

1.2.1 A. Naive Bayes

- Achieved the **highest accuracy** in both 5-fold (0.6808) and 10-fold (0.6964).
- Precision, recall, and F1 scores were consistently the best among the three models.
- Performs well on this dataset due to its ability to handle conditional independence assumptions efficiently.

1.2.2 B. KNN (k = 5)

- Performed moderately across all metrics.
- Accuracy remained close to Naive Bayes but lower in both 5-fold (0.6792) and 10-fold (0.6696).
- Performance depends on distance metrics and suffers slightly due to small dataset size.

1.2.3 C. Decision Tree

- Showed the **lowest performance** among all models.
- Accuracy dropped to 0.63 (5-fold) and further to 0.5803 (10-fold).
- Likely **overfits** the training data, leading to weaker generalization.

1.3 3. Cross-Validation Comparison

1.3.1 5-Fold vs 10-Fold

- Results were slightly more stable and improved in the **10-fold** evaluation.
- 10-fold CV reduces variance because each model is trained on a larger portion of the dataset.
- Naive Bayes benefited the most from the 10-fold split.

1.4 4. Overall Conclusion

- **Naive Bayes is the best-performing classifier** for this dataset across all evaluation metrics.
- **KNN** provides acceptable performance but is outperformed by Naive Bayes.
- **Decision Tree** is not suitable for this dataset due to lower accuracy and overfitting issues.
- **10-fold cross validation** provides more reliable results than 5-fold for small datasets.

[ ]:

```
```